网卡监视报警程序使用说明文档

文档编号： LINX_<NSQ>_<TS>_<HLD>_1.0

文档名称： 网卡状态实时采集和报警程序使用说明文档

文档类别： <使用说明>

密　　级： 公司内部

版本信息： 1.0

建立日期： 2009-05-04

创 建 人： 北京凝思科技有限公司研发部

审 核 者： 陈斌

批 准 人： 彭志航

批准日期： 2009-05-04

保 管 人： 武萌

存放位置： svn://.....

编辑软件： OpenOffice 3.0

文档修订记录

VER	REV	修订人	修订日期	简要说明	批准人	批准日期
V1.0	<C>	<胡剑>	09 年 5 月 4 日	<初次创建>	彭志航	09 年 5 月 4 日

VER（版本编号）：V——版本编号；R——修订编号。
REV（修订状态）：C——创建；A——增加；M——修改；D——删除。

文档审批记录

版本号	审批人	角色	审批日期	签字	备注
V1.0	彭志航	部门经理	09 年 5 月 4 日		

内容目录

1 软件概述 3
2 软件使用明细.....3
2.1 配置文件使用说明.....3
2.2 头文件使用说明.....4
2.3 网卡状态获取应用程序 sys_nicmonitor 使用说明.....7
2.4 读取共享内存接口函数使用说明.....7
3 其他事项说明.....8

1 软件概述

网卡状态实时采集和报警程序运行在D-5000的每个节点，该程序为D-5000平台的底层程序，独立于操作系统，用户名为d5000。网卡状态采集和报警程序定时向内核驱动请求网卡状态和信息，由网卡驱动接口提取网卡寄存器值，然后把相关信息和统计的状态返回到用户空间，将本节点各个网卡（包括虚拟网卡）的状态，写入本地的网卡状态共享内存，应包含时标信息；应提供变化数据写入和周期全数据写入两种方式，周期全数据写入时间可调，暂定为5分钟，如检测到网卡状态变化应实时写入。

网卡状态实时采集和报警程序当发现网卡故障、流量异常和网卡切换时，实时发送UDP报警事件报文，每个事件连续发送3次。

2 软件使用明细

2.1 配置文件使用说明

程序运行采用配置文件，配置文件路径为

/home/d5000/project/conf/sys_netcard_conf.txt。

配置文件内容如下：

```
domain              10                                  //程序运行域
serv                01                                  //网卡报警服务标识
event               2                                   //网卡报警事件ID标识
```

```
udpport            15000                              //网卡报警关播地址
write_interval        300          //周期写入共享内存的时间间隔，单位为秒
flow_ interval         60                    //获取流量时间间隔，单位：秒
flow_peak             100  //单块网卡的每秒最大流量，单位：兆字节每秒(Mb/s)
flow_limit             30                              //设定警报峰值的百分比
udp                  eth0       //发送广播包的网卡，通过此网卡获取广播地址
nic                  eth1                              //需要检测的网卡名称
nic                 eth2                    //如要监视多个网卡添加规则如此
nic                 eth3
```

2.2 头文件使用说明

公共头文件为：sys_netcard.h

其内容入下：

```
#ifndef _SYS_NETCARD_H
#define _SYS_NETCARD_H

#include <sys/socket.h>

#ifndef NIC_NAME_LEN
#define NIC_NAME_LEN        16
#endif

#define MAXNICNUM    32
#define SEM_PROJ_ID     's'

#ifdef __cplusplus
extern "C"
{
#endif
```

```
typedef   long       KEYID;

typedef struct  _msg_frame                  // message frame
{
   short  len;                             // message length
   short  seqno;                           // send sequence
   short  serv;                              // services ID    在配置文件中
   short  event;                           // event ID      在配置文件中
   unsigned char domain;                      // domain ID     在配置文件中
   unsigned char ctxt;                     // Context ID
   short  stid;                            // SOURCE task id
   short  dtid;                             // DESTINATION task ID
   unsigned char ver_coding;                // 版本号 + 编码
   unsigned char mes_type;                  // 帧类型
} MSG_FRAME, *LPMSG_FRAME;

typedef   struct  _net_info
{
   KEYID  ID;                             /*ID  国调保留 long */
   char   charname[NIC_NAME_LEN];            /*设备名称 */
   char   descr[128];                        /*设备中文描述 */
   struct sockaddr addr;                     /* IP 地址 */
   struct sockaddr broadaddr;              /*广播地址 */
   struct sockaddr netmask;                /*子网掩码 */
   struct sockaddr hwaddr;                 /* mac 地址 */
   int    time_stamp;                     /* 时标信息 精确到秒 读系统时钟*/
   short  flags;                             /* 状态标记 */
   int    mtu;                            /* 最大传输单元*/
   int    tx_queue_len;                      /* 传输队列长度 */
   int average_flow                       /* 网卡平均流量, 单位字节每秒*/
   unsigned long long rx_packets;          /* 接收包的数量 */
```

```
    unsigned long long tx_packets;      /* 发送包的数量 */
    unsigned long long rx_bytes;        /* 收到的总比特数 */
    unsigned long long tx_bytes;        /* 发送的总比特数 */
    unsigned long rx_errors;            /* 收到出错包数 */
    unsigned long tx_errors;            /* 发送包出错数 */
    unsigned long rx_dropped;           /* 因接收缓存不足丢包数 */
    unsigned long tx_dropped;           /* 因发送缓存不足丢包数 */
    unsigned long rx_multicast;         /* 组播包数 */
    unsigned long collisions;           /* 发生碰撞次数*/
    unsigned long rx_fifo_errors;       /* 管道空间溢出次数 */
    unsigned long tx_carrier_errors;
    unsigned long tx_fifo_errors;
}NETCARD_INFO, *LPNETCARD_INFO;

/* 状态标记 1:网卡故障 2:网卡恢复 3:网卡切换 4:流量异常。*/
#define NETCARD_ALARM_FAULT     1
#define NETCARD_ALARM_RESUME    2
#define NETCARD_ALARM_SWITCH    3
#define NETCARD_ALARM_ABNORMAL  4
typedef struct  _sys_netcard_alarm
{
    char fault_devname[NIC_NAME_LEN];     /*故障设备名称，主机名+网卡名 */
    char switch_devname[NIC_NAME_LEN];    /*切换设备名称 */
    short  flags;                         /* 状态标记 1:网卡故障 2:网卡恢复
                                             3:网卡切换 4:流量异常。*/
    short retrytimes;                     /* 0, 1, 2 重发次数 */
} SYS_NETCARD_ALARM, *LPSYS_NETCARD_ALARM;

typedef struct  _d5000_nic_alarm
{
    MSG_FRAME           tMsgFrame ;
    SYS_NETCARD_ALARM       tSysNetcardAlarm ;
```

```
}D5000_NIC_ALARM, *LPD5000_NIC_ALARM;


#ifdef __cplusplus
}
#endif
#endif
```

2.3 网卡状态获取应用程序 sys_nicmonitor 使用说明

使用此软件的方法是：

1. 以 d5000 用户登录，以下步骤若未特别指明用户身份默认均以 d5000 用户身份运行；
2. 将可执行文件 sys_nicmonitor 拷贝至 d5000 用户家目录的 bin 目录下；
3. 对可执行文件 sys_nicmonitor 执行：chmod u+s sys_nicmonitor 操作；
4. 将该软件的配置文件拷贝至 d5000 用户家目录的 conf/nic/目录下；
5. 以 root 身份编辑/etc/inittab 文件，在文件末尾加上一行：
   linx:2345:respawn:/home/d5000/.../sys_nicmonitor，该行最后一个冒号之后部分为存放可执行文件 sys_nicmonitor 的绝对路径，保存退出；
6. 以 root 身份执行命令：init q
7. 启动该软件后，使用 ps ax|grep sys_nicmonitor 查看一下 sys_nicmonitor 进程是否存在，若不存在则需查看日志信息，配置是否有问题。确定配置没有问题后再启动软件。

2.4 读取共享内存接口函数使用说明

本程序提供读取共享内存接口函数，并以动态库的形式提供。动态库为：libnic_shm.so。客户端编译程序时指定连接此库即可。

使用此接口函数方法是：

客户端程序调用 int get_nic_info(char *nic_name， NETCARD_INFO *net_info)开始读取共享内存数据。客户端程序需提供两个参数：nic_name 为网卡名，net_info 为存储网卡信息的结构体。该函数实现功能为：将所给网卡的全部信息写入 net_info 结构体中。该函数成功返回 0，失败则返回－1。

具体使用方法如下：

软件提供的动态库代码存放在/home/d5000/src/...目录下，目录里 nicinfo_shm.c 即为动态库的实现函数，nicinfo_shm.h 为您调用函数时要包含的头文件，sys_netcard.h 是公共头文件，也需作处理，getmem.c 是测试用例。

具体使用方法如下：

1，cd 到动态库代码存放的/home/d5000/src/...目录下，使用‘make’命令，此时生成动态库：libnic_shm.so；

2 ，将库拷贝至/home/d5000/src/lib/下或者用户指定的其他目录下；

3，编译程序时需指定头文件的位置，指定库的位置和库名，如测试用例 getmem.c 就在当前目录下则编译格式为：g++ －L . -lnic_shm -o getmem getmem.c

4，编译完成后 getmem 可直接执行，如./getmem eth0；

5 ,动态库的日志文件存放在 d5000 用户家目录的 var/log/netcard/目录下,文件名称是 dynamic_lib.log.

3 其他事项说明

1. 该软件所监视的网卡由正常变为异常，或由异常转为正常时均连续发送 UDP 报文 3 次，并将异常的网卡信息写入共享内存。
2. 根据客户提供的文档中要求，发送 UDP 报文的网卡名要求统一使用节点名（即主机名）加网卡名，而用来存储网卡名的空间只有 64 字节（该项定义在头文件中，为宏 NIC_NAME_LEN），鉴于此情况，本软件设定运行该软件的主机名字符窜长度加上网卡的字符窜长度最长不超过 64 个字节。
3. 根据 bondnig 模式只有 7 种，以及从实际应用出发，本软件限制每台运行该软件的服务器配置 bonding 的个数（bond0,bond1,bond2.....）最多不超过 10 个。
4. 该软件最多支持监视网卡数目定义在头文件中，为宏 MAXNICNUM。
5. 该软件提供的网卡信息中的平均流量在两种情况下有效：（1）将所监视的网卡信

息全部写入共享内存的时间周期到达时，写共享内存，此时各个网卡的平均流量值为这个时间周期内（配置文件指定的写周期 write_interval）的平均流量；（2）当网卡流量发生异常时，更新流量异常的网卡信息时所写入的网卡平均流量为配置文件中指定的计算流量时间间隔内（flow_interval）的平均流量。其它情况下读出的网卡平均流量均无效。

6. 该软件只支持 bonding 模式为 1（active-backup）的时候发生网卡切换时发送异常警报信息。Bonding 其它模式均不发生网卡切换。

7. 该软件计算网卡流量时，若 bonding 模式不为 1，且网卡为 bond？，则该网卡带宽为其各个备网卡带宽之和；若 bonding 模式为 1，且网卡为 bond？，则该网卡带宽为当前工作网卡带宽。

8. 公共头文件中 KEYID 指明由国调保留，故在此暂设为 0，不做他用。

9. 公共都文件中 descr[128]，指明设备中文描述，在此用来描述运行该软件的机器的 bonding 信息，描述各个 bond？下所管理的网卡。例如 bond0：eth0 eth1 bond1：eth2 eth3，表示 eth0 和 eth1 从属于 bond0，eth2 和 eth3 从属于 bond1。

10. 报警信息通过绑定到设备发送的意思是：获取该网卡设备的广播地址用来发送报文，若无此网卡设备或者此网卡设备没有广播地址则程序退出，记录日志出错信息为无效的广播设备。建议写配置文件之前使用/sbin/ifconfig 命令查看一下设备信息，选取已存在且已有广播地址的设备。

11. 该软件退出时，写有网卡信息的共享内存文件（shm_sys_netcard_info）会被删除，此时读取共享内存的应用程序将读不出数据。